Combining Syntax and Ontologies for Information Extraction

Amalia Todirascu, Laurent Romary, Dalila Bekhouche

Combining Syntax & Ontologies for Information Extraction

Amalia Todirascu*, Laurent Romary*, Dalila Bekhouche*

* INRIA Lorraine
LORIA, Campus scientifique BP 239, 54506 Vandoeuvre-lès-Nancy Cedex, France
{todirasc, romary, bekhouche}@loria.fr

Abstract

This paper presents an information extraction system, dedicated to message filtering for a specific domain (security systems). The paper focuses on a method for identifying domain-specific ontology elements (terms and concepts), using syntactic information and an existing domain ontology. The domain ontology is represented using description logics. The system uses description logics inference mechanisms to validate the candidate concepts.

1. Introduction

Information Extraction applications identify relevant entities in texts, for a given domain. They use shallow natural language processing techniques for identifying relevant data, and, in some cases, domain-specific knowledge to validate these entities. Most IE systems lack portability due to language-dependent linguistic resources and domain-dependent knowledge.

The IE systems' portability depends on the domain model (ontology) portability. Predefined or fixed ontologies are often incomplete and the costs of adapting them to another domain or application are enormous. Existing ontologies, like WordNet (Miller et al., 1990) and Corelex (Buitelaar, 1998), are useful for IE application on free texts but they do not contain domain-specific senses.

IE systems' portability might be increased using semi-automatically extracted ontologies. Several methods of extending existing ontologies have been proposed: statistical methods (Daille, 1996), as well as methods using logical inferences (Vilain, 1999). We used an inference-based method for extending ontologies to improve our system's portability.

Due to large amounts of processed texts, most IE systems propose candidate concepts among the noun phrases, noun-noun collocations (Heid, 2000), applying shallow Natural Language Processing (NLP) techniques: finite-state methods (Chanod, 1999) and simple pattern matching (Daille, 1996; Riloff et al., 1999). Some systems use domain-specific patterns to build a semantic representation from the syntactic structures (Riloff et al., 1997). Other methods extract domain-specific morphemes to filter the noun-to-noun and noun-to-adjective collocations (Heid, 2000). Special named entity recognisers have been proposed (Cunningham et al., 1996).

Most IE tools are highly language-dependent. It is difficult to adapt them for another language or domain. For this reason, we use linguistic tools and resources which are highly modular and which could be parameterized (the Lexical Tree Adjoining Grammar parser (Lopez, 1999)), for identifying candidate concepts.

2. Goal of the paper

The main goal of the paper is to present a methodology for extending a domain ontology with new concepts extracted from text. The paper focuses on the interface between syntax and semantics, in the context of a specific domain.

The method we propose uses partial syntactic structures provided by a Lexical Tree Adjoining Grammars (LTAG) parser to identify candidate concepts, as well as logical inferences from Description Logics, for knowledge representation. The method is tested with a specific application for filtering electronic messages about computer security problems. The corpus contains various errors (spelling, syntactic), wrong segmentations and computer commands. Due to these properties, a concept instance identification method based on partial parsing results is required for this corpus.

3. The architecture

The methodology proposed includes several steps in order to identify the key elements of the ontology: the concept instances, the relations between the concepts and the relations between the instances and the concepts. Concepts instances are identified among the results of partial parsing, while relations between concepts and between instances and the concepts requires domain-specific resources.

The system includes several modules for identifying domain concepts: the LTAG parser, the module handling the domain ontology and a module linking the syntax and semantics. The semantic representations, built from the syntactic structures and from some domain-specific resources (the semantic lexicon), will be validated by the domain ontology.

For our specific application, we developed modules for extracting lexicons or other resources from reference corpora (see figure 1). The reference corpus is used, together with the LTAG grammar and lexicon, to create a domain-specific lexicon (containing syntactic information). The corpus is also used by the human expert for designing the ontology.

We also built a module assigning concepts to each word, as well as assigning conceptual descriptions to grammar trees. The domain-specific lexicon is used by the LTAG parser to identify potential concept instances, among simple noun phrases and verbs.

4. Ontologies

Domain knowledge is a key element in an information extraction system. It is used to validate the entities

identified in the texts. Existing generic ontologies (WordNet (Miller et al., 1990), Corelex (Buitelaar 1998)) are useful for providing synonyms if the application is designed for free texts. Domain-specific ontologies often contain specific senses that would not be found in a general-purpose ontology.

To avoid the main drawback of domain-specific ontology, the low portability, statistical-based and inference-based methods for automatically extracting ontologies have been proposed. The methods identify candidate terms in texts, create classes of similar terms and identify relations between these classes.

Statistical methods group terms with similar contexts into classes associated with a unique concept (Assadi et al., 2000), (Daille, 1996). Relations between various classes are identified by interpreting syntactic structures (Capponi et al., 2000) or using statistical information (Bouaud et al, 2000). Statistical methods require large, stable, training corpora to extract the classes, as well as human experts for validating and interpreting the results.

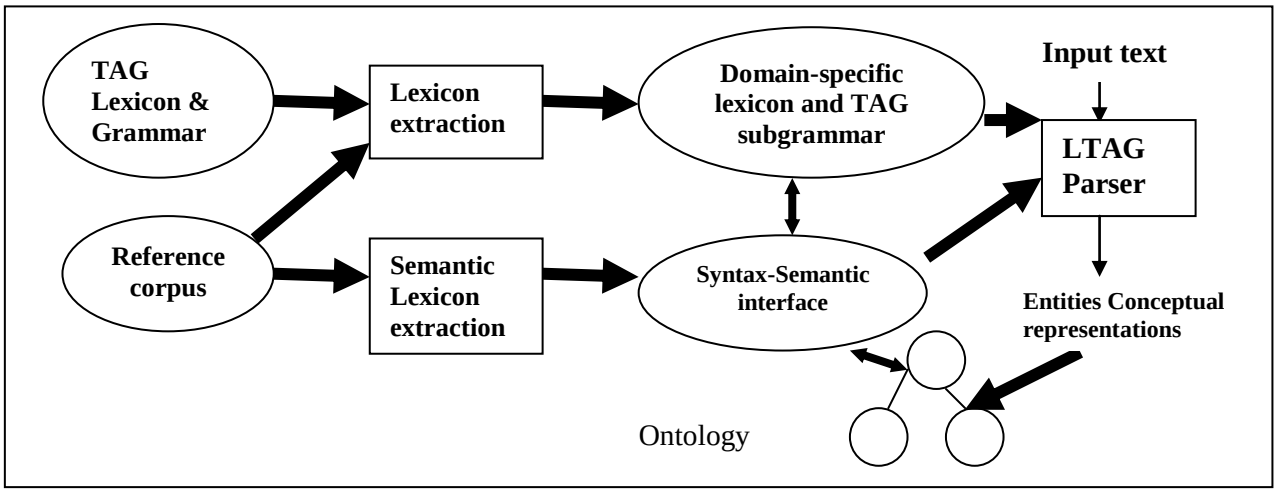

Figure 1. System architecture

Inference-based methods use semi-automatic methods to check the validity of the existing knowledge. New concepts, deduced by inference rules, are added to the domain hierarchy if they are coherent with the existing knowledge (Todirascu, 2001). Relations are identified using syntactic knowledge, as subcategorisation (Capponi et al., 2000). Concept overgeneration and the cost of checking knowledge incoherence and inconsistency are the main drawbacks of these approaches. Several formalisms have been proposed to avoid these problems, and among them description logics are very promising for representing domain knowledge.

4.1. Description Logics

Description Logics are knowledge representation formalisms derived from semantic nets, that provide well-defined syntax and semantics. They also have features of object-oriented systems, frame-based systems and modal logics.

DLs provide a hierarchical organization of knowledge structuring it on a conceptual level (the T-Box), describing abstract classes of objects relevant for domain modeling, and an assertional level (the A-Box), containing the instances of the classes. The classes of objects (concepts) are described by their relations (named roles) with other concepts and their attributes (roles with atomic values).

4.1.1. Syntax and Semantics

The DL operators are inspired by first order logics:

DL Operator	**Logic operator**	**Semantics**
D = SOME R C	$\exists$x (xRC)	there is at least one instance of C related by the relation R
D = ALL R C	$\forall$x(xRC)	restricts the co-domain of the relation R
D = AND C1 C2	C1 $\wedge$ C2	Conjunction of conceptual descriptions
D = OR C1 C2	C1 $\vee$ C2	Disjunction of conceptual descriptions
C1 $\subseteq$ C2	C1 $\Rightarrow$ C2	Axiom: C1 are necessary conditions for C2
D= NOT C	$\neg$ C	the complement of the concept C
D = $\geq$n.R.C	$\exists$y1…yn (1 $\leq$i$\leq$n, R(x, yi)$\wedge$C(yi))	There are at least n objects of the concept C in relation R with D

Table 1: The DL operators

Using all these operators, or only a part of them, we define our knowledge items: the definition of concepts and roles *ALC (Attributive Language with Complements)* (using atomic concept names and roles, the SOME, ALL, AND, OR, NOT operators, concept axioms), the possibility of handling transitive roles (R+), of inverse roles (I), of role hierarchy (H), of attributes (f) or numeric constraints.

A few examples of the DL commands are given below. CN is a concept name, C is a conceptual description (any combination of AND, SOME, NOT, ALL operators). The DL commands are KRSS-like (Patel-Schneider et Swartout, 1993), (Baader et Hollunder, 1991):

1) (define-concept CN C) - defines a new concept as a conceptual description;

2) (instance IN C) - defines an instance of a given concept;

3) (implies C1 C2) - introduces a new concept axiom, defining necessary conditions C1 for the conceptual description C2;

DLs are deterministic fragments of first order logics. They provide decidable algorithms for coherence and consistency checking. DLs propose logical mechanisms to identify concept subsumption, instance retrieval and role paths relating concepts. Classification is a partial ordering of the concept hierarchy due to the subsumption relation.

Some example of commands (using KRSS-like (Baader & all, 1991) syntax) are:

(concept-subsumes? C1 C2) tests if C1 subsumes C2

(concept-parents C) retrieves the direct ancestors of the concept C

(concept-children C) retrieves the children of C

(classify-tbox) computes all the subsumption relations between all the concepts defined in the T-Box

(concept-instances C) retrieves the instances of the concept C

4.2. Description Logics for IE Systems

The role of domain knowledge in an IE system is to validate the semantic representation of the potentially relevant entities identified in the text using NLP techniques. These entities could be used for adding new concepts to the existing ontology. Most IE systems use shallow NLP techniques and some candidate entities might not have a valid semantic interpretation. IE systems might use some implicit knowledge, the case of hyponymy/hyperonymy relations being just an example.

Unlike frame-based systems, DLs deal with semi-structured or incomplete data. There is no requirement to explicitly define some values as concept instances. Unlike frame-based systems, default values are not used by DLs. Some role fillers are left unspecified as in the following example.

```
(define-concept Administrator
    (and Person
       (some hasHandleType OSystem)
       (some hasName Name)))
(define-primitive-concept Name)
(define-primitive-concept OSystem)
(instance harry1
    (and Administrator
          (some hasHandleType Linux)))
```

In this example, we illustrate that implicit definitions are supported by DLs (**Linux** is not defined explicitly as an instance or a subconcept of the concept **OSystem**). The filler of the role **hasName** is not provided.

These properties are interesting for our application, while errors are possible, and the domain knowledge is incomplete.

Hyperonymy or hyponymy are handled by subsumptions between domain concepts. For example, if a candidate concept is identified in the text as

```
(instance x
    (and PC
    (some hasOperatingSystem Linux)))
(define-concept PCcomputer
    (and computer (some hasType PC)))
```

x is also an instance of the concept **computer**.

```
(instance y
    (and File (some hasOwner Root)))
(define-concept File
    (and Object
       (some hasName Name)
       (some hasPartOf OSystem)
       (some hasOwner User)))
(define-concept Root
    (and User
          (some hasRight ilimited)))
```

y is an instance of the concept **File** and it is related to the concept **Root**, which is also a **User**.

Semantic networks or frame-based systems provide hyponymy/hyperonymy handling, but the possibility of expressing negations is an argument in the favor of using description logics as knowledge representation. However, the interpretation of negation as the complement of the concept is not always very useful in a NLP application.

4.3. Functionalities

To choose the best DL classifier for our application, we identify precisely the functionalities that we require for representing knowledge.

We need to define concepts, roles and attributes to describe domain ontology. This is a feature, as well as axiom definitions, provided by all existing DL systems. Transitive roles are necessary in computing path roles between various concepts. Inverse roles must be avoided due to cyclical definitions which makes computing processes undecidable.

The input text contains proper nouns, data, person and organization names, which are represented in Description Logics as instances. Reasoning with instances is a necessary feature. We chose RACER (Haarslev, Muller, 2001), which is one of the few classifiers featuring such reasoning. It also has an XML-like representation of the ontology. Our aim is to build portable ontologies, and we intend to adopt a standard format for ontology representation – the Ontology Inference Layer (OIL) (Fensel et al., 2000).

4.3.1. The Ontology

The ontology is designed by a human expert. From the reference corpus, the human expert extracts a list of the most frequent words, except the functional words (the determiners, the pronouns, the auxiliaries). The list of nouns and verbs (considered as main candidates to identify primitive concepts) are evaluated by the human expert who assigns to each word a number (5-very relevant, 1-not very relevant) representing relevance of the word to the security domain. Some examples of words and their relevance:

Verb	Mark
Access ... to	2
Administer	2
Alter	4
Bypass	5

Verb	Mark
Change	1
Compromise	5
Configure	2
Connect ... to	3

Table 3: Verbs and their relevance to domain

From these words, the experts created a set of 35 primitive concepts and a set of concept axioms and roles. The concept candidates are added to the ontology if the ontology coherence is not violated.

The human expert also selects concept instances (the terms which are associated with each primitive concept). The expert asks the Racer classifier to test the coherence of the knowledge base.

5. NLP tools and resources

This section presents a short description of NLP resources and tools, as well as of the small ontology used for our application. The system is still under development. We developed the various NLP tools and resources, the modules extracting the lexicons and we built the initial ontology.

5.1. The Resources

5.1.1. The Reference corpus

The reference corpus used for generating the lexicon and for creating the ontology is a collection of phrases extracted from e-mail messages. It contains about 50,000 tokens (4039 word forms). The list of the most significant words, used also to design the domain concept hierarchy, was extracted from this reference corpus.

The corpus contains a lot of abbreviations, organization, system, program and function names (UNIX functions, constants, variable names), patterns introducing the content of a dialogue ("X wrote:"), DOS or UNIX commands, which require special preprocessing modules using finite state automata designed for entity recognition.

The corpus contains syntax or spelling errors, so we applied robust, fault-tolerant, shallow NLP techniques for identifying potential concept instances. Another major problem of the corpus is that the sentences are not very well delimited, which is one of the sources of tagging errors.

5.1.2. The lexicon

We used the reference corpus for building a lexicon. The lexicon is encoded using a subset of XML dedicated to represent TAG grammars: TAGML (Tree Adjoining Grammar Markup Language) (Bonhomme et al., 2000). The English LTAG lexicon (511 lexicon entries), available in TAGML format, was incomplete for our security domain. We built a module that semi-automatically created the lexicon from the reference corpus and from the existing English lexicon. For this, we used the TreeTagger POS tagger (Schmid, 1994), which annotates words with their lexical category and extracts the lemmas. Noun and adjective entries are generated automatically from the existing lexicon. Verbs must be added manually.

The list of POS tags provided by the TreeTagger is compatible with the lexical categories encoded into the lexicon.

The POS-tagging results must be validated by a human expert. We detected a set of errors including: spelling errors, POS tagging errors due to bad sentence construction and Unix commands (which must be preprocessed by a separate module) which must be deleted from the list of entries:

Lexical category	Occurrences	Errors
Nouns, Proper nouns	2991	10.04%
Verbs	1162	2.32 %
Adjectives	796	0.31 %
Adverbes	381	0.07%
Prepositions	105	4.46%

Table 2: The results of POS tagging and tagging errors

The resulting lexicon has 3000 entries, distributed as follows: 1400 nouns, 787 adjectives, 300 verbs, 105 prepositions and 379 adverbs. The lexicon also contains a set of syntagms (“one of”, “kind of”, “number of”).

5.1.3. The grammar

The initial English LTAG grammar (Joshi, 1987) contains about 420 elementary trees. The grammar is represented, as the lexicon is, in TAGML format. We concentrate at this stage on simple noun phrases (without relative clauses) and simple verb phrases, which will be the candidate concepts.

LTAG grammars have an interesting property, making them suitable for our application: the extended domain locality. It creates a context of each word, formed by a set of elementary trees. This property is used to avoid some candidate trees and to select a small subgrammar.

The local grammar built for this application consists of the noun phrase trees (trees associated with nouns, nouns and adjectives that might modify nouns, past participle verbs playing the role of the modifiers, comparative adverbs and adjectives), proper nouns and prepositions relating two nouns. We do not yet include verb trees handling long-distance dependency trees or relative clauses in the local grammar.

5.2. The parser

The parser we use is a current version of Lopez's parser (1999) that is modular and supports TAGML-like input and output. We chose this parser due to its ability to produce partial parsing results, and due to the existing TAG grammars (being available in TAGML format for French and for English). It is able to identify candidate terms, even if an error (e.g. syntactic or spelling) occurs. It is implemented in Java. The parser loads only a part of the lexicon and of the grammar (the entries associated with the current sentence).

TAG parsers combine the set of elementary trees associated with input words, using two operations: adjunction and substitution. TAG parsers produce as output a set of derived trees (which represent the syntactic structures identified in the text) as well as a set of derivation trees. Derivation trees are usually used to

generate a dependency tree: a semantic representation. TAG parsing provides a large set of derived and derivation trees. The derivation trees are used to create a semantic representation and this representation is validated by the domain ontology.

6. Linking syntax and domain ontology

This section presents in detail the relations between ontology concepts and roles to lexicons, grammars and parsing output. The interface between syntax and semantics consists of a list of pairs (lemma, concept), a set of conceptual descriptions built for each Elementary tree and an algorithm building conceptual descriptions from derivation trees.

6.1. The Semantic Lexicon

The LTAG lexicon entries contain references to lemmas associated with the current word, and each lemma is associated with a set of elementary trees and co-anchors. While each lemma might be associated with a different sense, we should obviously associate a different concept with each lemma.

Nouns are associated with one or several domain concepts. They are also associated with an entry describing the situation of a noun playing the role of a modifier. In this case the semantic description is a DL formula as (some hasMod Concept). Adjectives might be modifiers, but could also be predicative (and the DL representation is similar to verbal entries).

The verbs are associated with the concept and have some constraints on their arguments' type.

Example.
- for the noun 'system':
<sem concept="system" lemma="system"/>
- for the adjective 'main':
<sem concept="(some hasMod main)" lemma="main"/>
- for the verb 'connect':

```
<sem concept="connect" lemma="connect">
      <constr arg0="Substitution" address = "1"/>
      <constr arg1="Substitution" address = "3"/>
</sem>
```

Some lexicon entries are highly ambiguous: prepositions might represent several domain-specific relations or general relations (type, possession). For example, the preposition 'of' could be represented as:

<sem concept="(some hasType)" lemma="of"/>
<sem concept="(some hasPoss)" lemma="of"/>

If we define a function Sem taking as argument a lemma and returning a DL concept as a value:

Sem(lemma)=Concept ∧ Constraint1 ∧ …
∧ ConstraintJ

The concepts will be associated with words using a method similar to (Riloff et Sheperd, 1997), starting from a set of seed words and concepts (words extracted from the list of the most relevant words).

6.2. Elementary Trees

The elementary trees are related to the lemmas associated with each word. The semantic representation associated with each lemma is a primary concept and a set of constraints. We define an algorithm building a DL conceptual description for each elementary tree and the lemma associated to it.

Sem(ElementaryTree) = Sem(lemma), if no substitution or adjunction is found in the tree;

Sem(ElementaryTree) = (and Sem(lemma) (some hasSubstitution A)) ∧ (implies (some hasSubstition A) (some argi A)), if a substitution is found in the tree and A is a generic concept.

Sem(ElementaryTree) = (some hasAdjunction Sem(lemma)).

6.3. Derivation Trees

The derivation tree encodes all the operations (substitution, adjunction) applied for building a tree from elementary trees and other derivation trees. We define an algorithm for extracting complex representations from derivation trees.

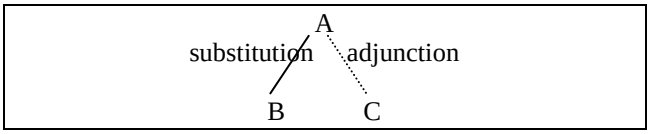

Figure 1. A derivation tree where A, B and C are also elementary or derivation tree

The algorithm building the DL representation for the tree from the fig. 1:

Sem(Tree) = (and Sem(A) (Some hasSubstitution Sem(B)) Sem(C)) ∧ (constraints B)

where Sem(A), Sem(B) and Sem(C) are defined recursively, if B or C are derivation trees, and as in the previous subsection if A is elementary tree. (constraints B) represents the information related to the arguments.

6.3.1. Some Examples

The phrase "the root passwords" is parsed by the LTAG parser, and the results are two derivation trees (figure 2) The concepts associated with each lemma are:

Sem(root) = Root
Sem(root) = (some hasMod Root)
Sem(password) = Password
Sem(the) = (some hasDef definite)

The conceptual representation associated with each derivation tree is

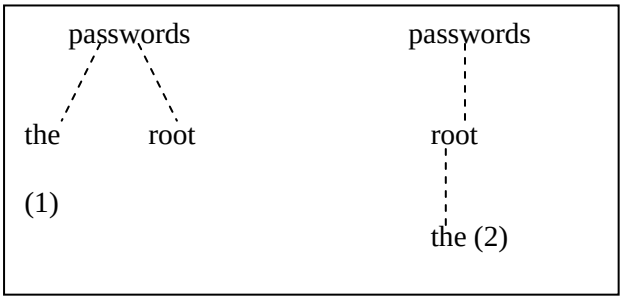

Fig. 2 Derivation trees for "the root password"

Sem(1) = (and Password (some hasMod Root)(some hasDef definite))

Sem(2) = (and Password (some hasMod (and Root (some hasDef definite))))

The phrase "The hacker connected to the server" results into the following derivation trees (fig.3).

The conceptual descriptions associated with each lemma are:

Sem(connected_to) = Connect ∧ (implies (some hasSubst A) (some arg0 A)) ∧ (implies (some hasSubst B))

Sem(hacker) = Hacker

Sem(server) = Server

Sem(the) = (some hasDefine Defined)

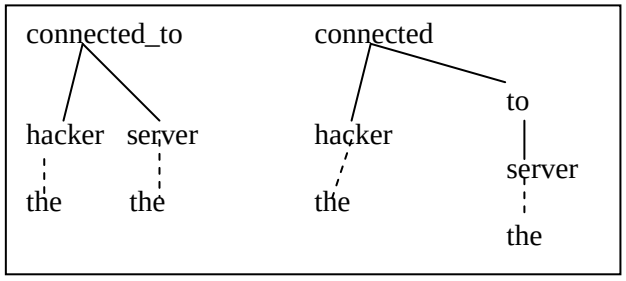

Fig. 3 Derivation trees for "the hacker connected to the server"

The representations of the trees substituted to the elementary tree anchored by 'connected' are:

```
(and Hacker (some hasDefine Defined))
(and Server (some hasDefine Defined))
```

The representation associated with the elementary tree anchored by 'connected' is the set of axioms:

```
(and Connect (some hasSubst A)(some hasSubst B))
(implies (some hasSubst A)(some arg0 A))
(implies (some hasSubst B)(some arg1 B))
```

In the derivation tree we found the values for A and B:

```
(implies A (and hacker (some hasDefine Defined)))
(implies B (and server (some hasDefine Defined)))
```

The concept associated with the derivation tree:

```
(and Connect
     (some arg0 (and hacker (some hasDefine Defined)))
     (some arg1 (and server (some hasDefine Defined))))
```

is satisfiable by the existing knowledge.

7. Extending the ontology

We use the results of partial parsing to identify a set of relevant entities. Some of them might be added to the existing ontology, after a human expert validates them.

The corpus contains a lot of errors and a complete parsing will never result in a correct syntactic structure.

Example from corpus:

```
Remains the question how are you going to
run DOS programs if the first thing that
the computer does after the password
protected BIOS is coming up with a lilo
prompt for a password ?
```

The entities identified in the text are : `run, DOS, the first thing, the computer, the password protected BIOS, a lilo prompt for a password.`

We obtained the following conceptual descriptions:

```
(and run
     (some has arg1
            (and program
                 (some hasType DOS))))
(and computer (some hasdef Definite))
(and entity
     (some hasname BIOS)
     (some hasProp
            (and password
            (some hasType protected)
            (some hasdef definite)))
(and prompt
     (some hasType lilo)
     (some hasdef indefinite)
     (some hasPurpose
            (and password
               (some hasdef indefinite))))
```

These representation are validated by the existing ontology. We might add a new concept *run* with an argument restricted to the type *program*.

8. Conclusion and further work

The paper presents a methodology for extracting concepts from texts, using partial syntactic analysis and domain knowledge. It focuses on the relation between the syntax and the domain-specific knowledge, in order to build candidate concepts for the domain ontology. We intend to use a meta-grammar for generating automatically the elementary trees. We will extend this meta-grammar to generate conceptual descriptions together with the elementary trees. We will also develop a method for generating verbs lexicon entries automatically.

Acknowledgements. Amalia Todirascu worked on this project as a post-doctoral fellowship in LORIA, sponsored by the European Research Consortium for Informatics and Mathematics (ERCIM) group.

9. References

Assadi, H., Bourigault, D., 2000, Analyse syntaxique et statistique pour la construction d'ontologies à partir des textes. In J.Charlet, M.Zacklad, G.Kassel, D.Bourigault (eds.) - Ingénierie des connaissances - Evolutions récentes et nouveaux défis, Eyrolles Publishing House, pp. 243-256.

Baader, F., Hollunder, B., 1991, A Terminological Knowledge Representation Systems with Complete Inference Algorithms, Proceedings of the Workshop on Processing Declarative Knowledge.

Bonhomme, P., Lopez, P, 2000, TAGML: XML encoding of Resources for Lexicalized Tree Adjoining Grammars. In Proceedings of LREC 2000, Athens.

Bouaud, J., Habert, B., Nazarenko, A., Zweigenbaum, P., 2000, Regroupements issus de dépendances syntaxiques sur un corpus de spécialité: catégorisation et confrontation à deux conceptualisations du domaine. In J.Charlet, M.Zacklad, G.Kassel, D.Bourigault (eds.) - Ingénierie des connaissances - Evolutions récentes et

nouveaux défis, Eyrolles Publishing House, pp. 243-256.

Buitelaar, P., 1998, CORELEX: Systematic Polysemy and Underspecification, Ph.D. thesis, Brandeis University, Department of Computer Science

Capponi, N., Toussaint, Y., 2000, Interprétation de classes de termes par généralisation de structures prédicat-argument. In J.Charlet, M.Zacklad, G.Kassel, D.Bourigault (eds.), Ingénierie des connaissances - Evolutions récentes et nouveaux défis, Eyrolles Publishing House, pp. 337-356.

Chanod J.P., 1999, Natural Language Processing and Digital Libraries. In M.T.Pazienza (ed.), Information Extraction, Springer-Verlag, LNAI 1714, pp.17-31.

Cunningham, H., Wilks, Y., Gaizauskas, R.J., 1996, New Methods, Current Trends and Software Infrastructure for NLP. In Proceedings of the conference on New Methods in Natural Language Processing (NeMLaP-2), Bilkent University, Turkey, 1996, pp.1-12.

Daille, B., 1996, Study and Implementation of Combined Techniques for Automatic Extraction of Terminology. In J.Klavans, P.Resnik (eds.), The Balancing Act - Combining Symbolic and Statistical Approaches to Language, MIT Press, pp. 49-66.

Fensel, D. et al., 2000, OIL in a nutshell. In R. Dieng et al. (eds.), Knowledge Acquisition, Modeling, and Management, Proceedings of the European Knowledge Acquisition Conference (EKAW-2000), Lecture Notes in Artificial Intelligence, LNAI, Springer-Verlag.

Heid, U., 2000, A linguistic bootstrapping approach to the extraction of term candidates from German text. In Terminology, pp 161-180.

Haarslev V., Muller R, 2001, Description of the RACER System and its Applications. In Proceedings of the International Workshop on Description Logics (DL-2001), Stanford, USA, 1.-3. August 2001, pp. 132-141

Joshi A., 1987, An Introduction to Tree Adjoining Grammars. In Mathematics of Language, John Benjamins Publishing, Amsterdam/Philadelphia, 87-115.

Lopez, P., 1999, Robust Parsing with Lexicalized Tree Adjoining Grammars, Ph.D.Thesis, INRIA, Nancy, France.

Miller, G., Beckwith, R., Fellbaum, C., Gross, D., Miller K., 1990, Introduction to WordNet: An On-Line Lexical Database. In International Journal of Lexicography, 3(4), pp.302-312.

Patel-Schneider, P.F., Swartout B., 1993, Description logic specification from the KRSS effort, Technical Report, ARPA Knowledge Sharing Effort Project.

Riloff, E., Lorenzen, J., 1999, Extraction-based Text Categorization Generating Domain-Specific Role Relationships Automatically. In ed. T.Strzalkowski, Natural Language Information Retrieval, Kluwer Academic Publishers, pp. 167-196.

Riloff, E., Shepherd, J., 1997, A Corpus-Based Approach for Building Semantic Lexicons. In Proceedings of the Second Conference on Empirical Methods in Natural Language Processing.

Schmid, H., 1994, Probabilistic Part-of-Speech Tagging Using Decision Trees. In Proceedings of the International Conference on New Methods in Language Processing, Manchester, United Kingdom.

Todirascu, A., 2001, Semantic Indexing for Information Retrieval Systems, University Louis Pasteur of Strasbourg, Ph.D. Thesis.

Vilain, M., 1999, Inferential Information Extraction. In M.Pazienza (ed.), Information Extraction, LNAI 1714, Springer-Verlag, pp.95-119.